Table of Contents

Introduction

Brunel is very much setup to reprocess existing DSTs, but Boole was never engineered for that, and you need to do some work.

Prerequisites: SIM/XDST/XDIGI

The only file types which contain the information to re-run boole are SIM, XDIGI and XDST, the X stands for "extended" and means additional information is copied from the SIM to the DIGI. Other file formats do not contain the required information, and so you would need to restart a production.

Example files:

You can check if what you've got works by using these files:

```
from GaudiConf import IOHelper
#original data file
#IOHelper().inputFiles(["root://s35-02.grid.sara.nl:1094/pnfs/grid.sara.nl/data/lhcb/MC/Upgrade/X

#modified data file with a load of both Boole and Rec entries on it, completely broken because th
#IOHelper().inputFiles(["root://ccdcacli039.in2p3.fr:1094/pnfs/in2p3.fr/data/lhcb/user/t/thead/38

#example SIM file
from PRConfig import TestFileDB
#TestFileDB.test_file_db["boole.boole-mc11"].run()
```

Step 1: Cleaning your XDST

In your first job you need to remove everything that Brunel and Boole added to your XDST. To do that, run a job over each file which removes the rubbish, and writes a new file.

```
SetupProject #anything... any recent version of LHCb software will work!
```

gaudirun over your files with options like:

```
from Gaudi.Configuration import *
from Configurables import LHCbApp
from Configurables import EventNodeKiller

enk=EventNodeKiller("Reset_to_XSIM")
enk.Nodes=[
    #added by Brunel:
    "/Event/Trigger",
    "/Event/Rich",
    "/Event/Calo",
    "/Event/Muon",
    "/Event/Other",
    "/Event/Rec",
    "/Event/pRec",
    #added by Boole
    "/Event/Link/Raw",
    "/Event/DAQ",
    "/Event/pSim/Rich/DigitSummaries",
    "/Event/MC/TrackInfo",
    "/Event/MC/Muon",
    "/Event/MC/DigiHeader",
    #not always there, depends on the DataType and format
    "/Event/Link/Trig",
    "/Event/MC/Rich/DigitSummaries",
```

```
        "/Event/Prev/DAQ",
        "/Event/PrevPrev/DAQ",
        "/Event/Next/DAQ",
        "/Event/NextNext/DAQ"
        ]

LHCbApp().Simulation=True

ApplicationMgr().TopAlg+=[enk]

from GaudiConf import IOHelper

#note that you CANNOT use an InputCopyStream, because that might resurrect some of the things you

simwriter=OutputStream("simwriter")
IOHelper().outStream("Boole-Extended-Recreated.sim",simwriter)

#in this OutputStream we will maually recreate everything that should have been there only on the
#We could also use SimConf for this, probably, but then its options need to be set perfectly.
#By using OptItemList we can catch all possible sim file versions, or at least that is the idea.

simwriter.ItemList=[
    #MC content
    "/Event/MC/Header#1"
    ]

#Hits for all detectors including the upgrade ones, if you spot something missing, please add it
detcontent=[detector+"/Hits" for detector in ["OT","IT","TT","FT","UT","Muon","Prs","Spd","PuVeto

#extra things for the RICH
detcontent+=["Rich/OpticalPhotons","Rich/Segments","Rich/Tracks","Rich/Hits2MCRichOpticalPhotons"

#add to links and Event, for all possible spillover paths
for path in ["/Event/","/Event/Link/"]:
    for spill in ["","PrevPrev/","Prev/","Next/","NextNext/"]:
        for subdir,content in {"Gen/":["Header","BeamParameters","HepMCEvents","Collisions"],"MC/
            for item in content:
                simwriter.OptItemList.append((path+spill+subdir+item+"#1").replace("//","/"))

###add some input files here...
```

Call this file a ".sim", because that is really what it looks like.

Step 2: Checking your new Sim-like file

If everything worked, your extra locations should have disappeared from the file.

Take a look at the dump to see what's there now.

```
from Gaudi.Configuration import *
from Configurables import LHCbApp
from Configurables import RawEventDump
from Configurables import StoreExplorerAlg

LHCbApp().Simulation=True
StoreExplorerAlg().Load=True
RawEventDump().DumpData=False
LHCbApp().EvtMax=1


ApplicationMgr().TopAlg+=[StoreExplorerAlg(), RawEventDump()]

###add some input files here ...
```

Step 3: Rerunning Boole

Find the exact options used to create the first file, edit them for what it was you needed to change. Don't forget the very important DDDB, CondDB and Simulation flags needed. You can find most of this information from the book-keeping, which might point you to files in AppConfig where the options for the processing pretty much always sit.

-- RobLambert - 06 Mar 2014

This topic: LHCb > ReprocessFromBoole
Topic revision: r4 - 2014-03-11 - RobLambert